Two methods of pruning Benders' cuts and their application to the management of a gas portfolio

Laurent Pfeiffer, Romain Apparigliato, Sophie Auchapt

Two methods of pruning Benders' cuts and their application to the management of a gas portfolio

Laurent Pfeiffer, Romain Apparigliato, Sophie Auchapt

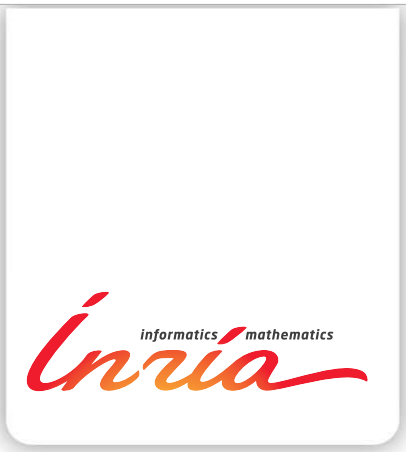
RESEARCH REPORT

N° 8133

November 2012

Project-Team Commands

ISSN 0249-6399 ISRN INRIA/RR--8133--FR+ENG



Two methods of pruning Benders' cuts and their application to the management of a gas portfolio

Laurent Pfeiffer*, Romain Apparigliato†, Sophie Auchapt‡

Project-Team Commands

Research Report n° 8133 — November 2012 — 23 pages

Abstract: In this article, we describe a gas portfolio management problem, which is solved with the SDDP (Stochastic Dual Dynamic Programming) algorithm. We present some improvements of this algorithm and focus on methods of pruning Benders' cuts, that is to say, methods of picking out the most relevant cuts among those which have been computed. Our *territory algorithm* allows a quick selection and a great reduction of the number of cuts. Our second method only deletes cuts which do not contribute to the approximation of the value function, thanks to a *test of usefulness*. Numerical results are presented.

Key-words: SDDP algorithm, stochastic programming, energy management, pruning methods, Benders' cuts.

This study was supported by the Centre of Expertise in Economics, Modelling, and Studies of GDF-Suez. The authors are grateful to Frédéric Bonnans and Yves Smeers for providing them useful advice.

* INRIA-Saclay and CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex, France (laurent.pfeiffer@polytechnique.edu)

† GDF-Suez, Centre of Expertise in Economics, Modelling, and Studies, 1 rue d'Astorg, 75392 Paris Cedex 08, France (romain.apparigliato@gdfsuez.com)

‡ GDF-Suez Centre of Expertise in Economics, Modelling, and Studies, 1 rue d'Astorg, 75392 Paris Cedex 08, France (sophie.auchapt@gdfsuez.com)

RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Deux méthodes d'élagage de coupes de Benders et leur application à la gestion d'un portefeuille gazier

Résumé : Dans cet article, nous décrivons un problème de gestion d'un portefeuille gazier, résolu avec l'algorithme SDDP (Stochastic Dual Dynamic Programming). Nous présentons quelques améliorations de cette algorithme et nous nous concentrons sur des méthodes d'élagage des coupes de Benders, c'est-à-dire, des méthodes pour sélectionner les coupes les plus pertinentes parmi celles déjà calculées. Notre *algorithme des territoires* permet une sélection rapide et une grande réduction du nombre de coupes. Notre seconde méthode ne supprime que les coupes qui ne contribuent pas à l'approximation de la fonction valeur, à l'aide d'un *test d'utilité*. Nous présentons des résultats numériques.

Mots-clés : Algorithme SDDP, programmation stochastique, gestion énergétique, méthodes d'élagage, coupes de Benders.

1 Introduction

The management of a gas portfolio is a major optimization problem which arises in an uncertain environment, since gas consumption and prices are random. The goal of the problem is to minimize the costs required to supply gas on different zones of consumption, each day of the year. The gas comes from delivery contracts and the demand can be regulated thanks to gas storages. Over one year, the total amount of gas bought on each contract is limited. This constraint, as well as the level of each storage, introduce a strong link between the decisions taken at each time step. This link will be taken into account thanks to a high-dimensional state variable.

In this article, we first give a simple formulation of this problem as a multi-stage stochastic linear problem and we give a description of the SDDP algorithm [4, 16] used to solve the problem. This well-known algorithm approximates the value function with the supremum of affine functions called Benders' cuts or optimality cuts. It has been mainly used in energy management and is still actively studied [8].

Secondly, we focus on methods of picking out the most relevant optimality cuts among those generated by the algorithm. Indeed, the first cuts which are computed may become obsolete after a few iterations. Though they do not contribute to the current approximation of the value function, they slow down the execution of the algorithm. Thus, there is a need to prune the set of optimality cuts.

This question has not been much adressed in the stochastic programming literature. In the regularized decomposition method described in [18], only the active cuts are selected, that is to say, those having a positive Lagrange multiplier associated at the last iteration of the algorithm. This method can be used for multi-stage problems if we consider the value functions associated with each node of the scenario tree, as in [19]. In our study, thanks to the stagewise indepence of the random variable, we only need to describe one value function for each time step. Therefore, the active cuts of the value function may differ from one realization of the demand to another and the method cannot be applied in this situation. The problem of cut selection has been recently studied in the framework of max-plus based approximation methods [12]. Similarly to SDDP, these methods approximate the value function of a nonlinear optimal control problem by a supremum of basis functions. In this case, the basis functions are quadratic functions. Pruning methods of these basis functions, different from ours, have been proposed in [15] and more recently in [14].

In a recent preprint [10], de Matos, Philpott, and Finardi describe a heuristic to select optimality cuts called the *Level 1 Dominance*. Before solving a linear program, they select the cuts to be used. Considering a set of points $(x_i)_{i=1,\dots,K}$, they only keep the cuts which dominate the others on at least one point x_i. Our *territory algorithm* uses the same heuristic to delete definitively cuts previously generated.[1] It fastens the solving of the problem and avoids exceeding the storage space capacity of the computer with a useless accumulation of cuts. However, the territory algorithm may delete some useful cuts. Our second method combines the territory algorithm with a *test of usefulness*, which examines if a given cut brings information. This method avoids a damaging of the approximation of the value function.

The aim of the article is to show the interest of pruning methods, thanks to simulations on a real-world problem. The territory algorithm allows a quick selection of the cuts and a reduction by a factor of 10 of the number of cuts. Although our second method is slow, it does not damage the approximation of the value function and shows that a reduction by a factor 5 is reachable. We observe that the time needed to realise forward passes is reduced by a factor 3 when the

[1] This algorithm was presented for the first time in 2007 at the ROADEF congress by David Game and Guillaume Le Roy (GDF-Suez).

territory algorithm is applied.

The organisation of the article is as follows. We describe the problem in section 2 and give a mathematical formulation of it in section 3. In section 4, we describe the SDDP algorithm and in section 5, we give a short review of computational improvements with precise references. In section 6, we describe the territory algorithm and the test of usefulness. In section 7, we provide some numerical results with a test case with a high-dimensional state space of dimension 19.

2 Gas portfolio management

2.1 General description of assets

In this part, we describe in general terms the gas assets of our problem. The gas network is composed of a set of balancing zones, representing geographic places of consumption. All the balancing zones are linked by pipes, and their gas inflows and outflows must be equal in order to satisfy the demand. To that purpose, various assets can be used.

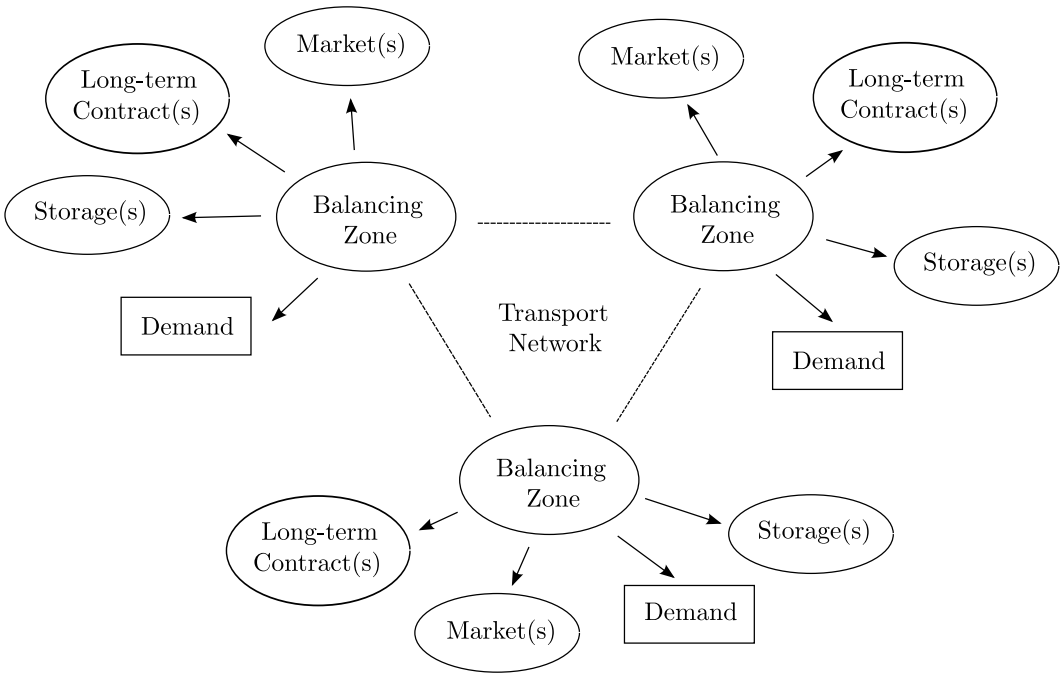


Figure 1: An example of Gas Network

Long-term (LT) delivery contracts They are usually signed for periods up to 20 or 30 years and include take-or-pay clauses (ToP): the buyer agrees to buy a minimum amount of gas at one or several delivery points during pre-specified periods of time (annual/quarterly/monthly constraints). This allows the seller to hedge his volume-risk and to pay off and secure huge investments in production fields. Minimal amounts of delivered gas per period are hard constraints: violating such a constraint would lead to financial penalization and unused gas would be lost. In exchange, to reduce his volume-risk, the seller guarantees a competitive price for the gas sold. These contracts can be seen as a swap of volume-risk against price-risk.

Storages In general, the gas available from LT contracts is not sufficient to face winter consumption peaks. On the contrary, when the demand is low in summer, take-or-pay clauses are such that the minimal gas deliveries are higher than the consumption. This lack of flexibility can be counterbalanced by storage facilities which deal with the strong seasonality of the demand: storages are filled in summer, when the consumption is low. In winter, they provide an additional resource to satisfy high loads, thus preventing calls to the market when prices are expected to get very high. Storages are mainly of two types: aquifers and salt caverns. Aquifers typically have high capacities but lower injection and extraction rates. They are used to smooth the load annual seasonality. By contrast, salt caverns have low capacity but higher delivery rates and may be used to cover peak loads or to perform daily arbitrages on the market.

Markets Spot market places are named "hubs". Physical hubs represent a big interconnection point between several gas assets. Virtual hubs are parts of the network on which exchange points could not be identified. We can mention an English virtual hub, the National Balancing Point, and the Belgium physical hub Zeebrugge.

2.2 Optimization variables and constraints

We describe in this part the gas portfolio problem. From the point of view of the operational gas portfolio manager, the main uncertainty comes from the consumption level. Its main objective is to balance supply and demand whatever the demand is. This is why we consider prices as deterministic and why we do not take into account the markets in the optimization problem.

2.2.1 Data

In this section, we use the following notations:

t	time index
T	horizon of the problem
S	set of storages s
A	set of contracts a
Z	set of balancing zones z
$SZ(z)$	set of storages linked with zone z
$AZ(z)$	set of contracts delivering gas on zone z
$I(a)$	set of quotas i of contract a
$\left[b^{a,i}, e^{a,i}\right]$	time period of quota i (which is associated to contract a)
d_t^z	demand of gas on zone z at time t
d_t	demand vector at time t.

The stochastic process associated with the demand and the decision process will be described in section 3. For the moment, we simply describe the variables and the constraints involved for one particular realization of the demand.

2.2.2 Variables

State variables The state variable is denoted by x_t. It is composed of the following assets:

$$\begin{array}{ll} x_t^s & \text{level of storage } s \text{ at time } t \\ x_t^{a,i} & \text{cumulated gas quantity used from the beginning of quota } i \\ & \text{of contract } a \text{ until time } t. \end{array} \tag{1}$$

Decision variables The decision variables are the following:

$$\begin{array}{ll} inj_t^s & \text{injection in each storage } s \text{ at time } t \\ wth_t^s & \text{withdrawal of each storage } s \text{ at time } t \\ q_t^a & \text{quantity bought on each contract } a \text{ at time } t \\ trans_t^{z,z'} & \text{quantity carried in the pipe from zone } z \text{ to zone } z' \text{ at time } t \\ g_t^{z+} & \text{positive slack on zone } z \text{ at time } t \\ g_t^{z-} & \text{negative slack on zone } z \text{ at time } t. \end{array} \tag{2}$$

Note that the slack variables will enable us to satisfy the supply-demand constraint.

2.2.3 Constraints

Bounded exchanges All the exchanges, withdrawals, injections, purchases and sells, are bounded. The following bounds are defined for each $t \in [0, T-1]$ as follows:

$$\begin{cases} 0 \leq inj_t^s \leq \overline{inj}_t^s & \forall s \in S, \\ 0 \leq wth_t^s \leq \overline{wth}_t^s & \forall s \in S, \\ \underline{q}_t^a \leq q_t^a \leq \overline{q}_t^a & \forall a \in A, \\ 0 \leq trans_t^{z,z'} \leq \overline{trans}_t^{z,z'} & \forall z, z' \in Z, \\ 0 \leq g_t^{z+} & \forall z \in Z, \\ 0 \leq g_t^{z-} & \forall z \in Z. \end{cases} \tag{3}$$

Note that the upper bounds on injections and withdrawals are approximate: they should depend on the storage level. Indeed, the higher the level of storage is, the higher the capacity of withdrawal is. However, this simplification enables us to keep the convexity of the problem, which is a key hypothesis of the SDDP algorithm. Let us mention that $\underline{q}_t^a$ and $\overline{q}_t^a$ have the same sign and can be negative (that is to say, we can sell gas on contracts for which $\overline{q}_t^a < 0$).

Supply-demand constraint On each zone $z \in Z$, at each time $t \in [0, T-1]$, the equilibrium constraint is given by

$$\sum_{s \in SZ(z)} (wth_t^s - inj_t^s) + \sum_{a \in AZ(z)} q_t^a + \sum_{\substack{z' \in Z \\ z' \neq z}} trans_t^{z',z} = d_t^z + g_t^{z+} - g_t^{z-} + \sum_{\substack{z' \in Z \\ z' \neq z}} trans_t^{z,z'}. \tag{4}$$

Dynamic of the state variables The evolution of the storages is described by

$$x_{t+1}^s = x_t^s + inj_t^s - wth_t^s, \quad \forall s \in S, \ \forall t \in [0, T-1], \tag{5}$$

and the initial level x_0^s is given, for all s in S. The evolution of the gas contracts is described by

$$x_{t+1}^{a,i} = x_t^{a,i} + q_t^a, \qquad \forall a \in A, \ \forall i \in I(a), \ \forall t \in [b^{a,i}, e^{a,i} - 1], \tag{6}$$

$$x_{b^{a,i}}^{a,i} = 0, \qquad \forall a \in A, \ \forall i \in I(a). \tag{7}$$

State constraints All the storage levels and all the cumulated levels on the long term contracts are upper and lower bounded as follows:

$$\begin{cases} \underline{x}_t^s \leq x_t^s \leq \overline{x}_t^s & \forall s \in S, \ \forall t \in [1, T], \\ \underline{x}_t^{a,i} \leq x_t^{a,i} \leq \overline{x}_t^{a,i} & \forall a \in A, \ \forall i \in I(a), \ \forall t \in [b^{a,i}, e^{a,i}]. \end{cases} \tag{8}$$

Originally, these bounds are given for some particular time steps. For the long term contracts, we only have a lower and an upper bound at the end of each quota, at time $e^{a,i}$, and the initial state at time $b^{a,i}$. Constraints (8) also include constraints (7). For a storage s, a lower bound is 0 and an upper bound is C^s, the capacity of the storage. Additionnal bounds (from physical constraints) are given by the storage manager.

All these constraints must be anticipated, because the exchanges of gas are themselves bounded. For instance, for storage s, at time $T-1$, the level must be at least equal to $\overline{x}_T^s - inj_s^{T-1}$, otherwise it is impossible to inject enough gas to satisfy the final constraint. We can compute minimal state constraints so that for each feasible state at time t, for each demand d_t, there exists a decision which leads to a feasible state at time $t+1$. We do not describe the computation of these state constraints. Note that the slack variables allow to satisfy the equilibrium constraint whatever the demand is.

2.2.4 Cost

The daily cost depends linearly on all the decision variables. The total cost associated with the scenario of demand $(d_0, .., d_{T-1})$ is

$$\sum_{t=0}^{T-1} \Big(\sum_{s\in S} (pwth_t^s.wth_t^s + pinj_t^s.inj_t^s) + \sum_{a\in A} pq_t^a.q_t^a + \sum_{z,z'\in Z} ptrans_t^{z,z'}.trans_t^{z,z'} + \sum_{z\in Z} (pg_t^{z+}.g_t^{z+} + pg_t^{z-}.g_t^{z-}) \Big), \tag{9}$$

where the unitary prices ($pinj_t^s$, $pwth_t^s$, pq_t^a, $ptrans_t^{z',z}$, pg_t^{z+}, and pg_t^{z-}) are known and deterministic. The daily cost does not depend on the state variables: this approximation is made to keep the convexity of the problem. Note that in practice, pg_t^{z+} and pg_t^{z-} are chosen very high, so that the slack variables are used ultimately.

3 Problem formulation

In this section, we give a mathematical formulation of the gas supply problem introduced in section 2. The problem is described by a multistage stochastic linear program with relatively complete recourse. For surveys on stochastic programming, we refer to [6, 21].

Variables and constraints Let us formalize the variables and the constraints involved at stage t of the global problem. We denote by:

- x_t, the state variable, described by (1)
- u_t, the control variable, which is the concatenation of inj_t^s, wth_t^s, q_t^a, $trans_t^{z,z'}$, g_t^{z+} and g_t^{z-}, described by (2)
- d_t, the demand of gas for the different zones.

We introduce the following notations for the constraints:

- $Bu_t \leq b_t$, the constraints on the control variable u_t, described by (3)
- $Eu_t = d_t$, the balance between supply and demand given by (4)

- $x_{t+1} = x_t + A_t u_t$, the dynamic of the state x_t given by (5) and (6)
- $x_{t+1} \in X_{t+1}$, the state constraint at time $t+1$, which must be anticipated at time t. X_{t+1} is a polyhedron described by (8).

The part of the cost function associated to stage t is denoted by $c_t u_t$. The parameter c_t is defined by (9).

Modelling of the demand For our problem, the uncertain data is the demand d_t of gas. It is described by a stagewise independent stochastic process: for two times t and t' with $0 \leq t < t' < T$, the real random variables d_t and $d_{t'}$ are independent. In the sequel, we use the same notation for a random variable and a particular realization of this variable. The distinction will be clear from the context. We consider a discrete process: for all t, d_t takes its value in

$$D_t = \{d_t^1, ..., d_t^i, ..., d_t^K\},$$

where K is the cardinal of D_t. The associated probabilities are denoted by $p_t^1, ..., p_t^i, ..., p_t^K$. We denote by $d_{[t]} = (d_0, ..., d_t)$ the history of the process until time t.

The set D_t is built from a set of consumption scenarios, which can be historical scenarios or scenarios obtained from a proper model. We assume that the demand has a normal distribution at each time step t. The mean value and the variance of the demand are estimated with the scenarios. We compute D_t from a quantized grid of the standard normal law[2].

Decision process The demand is discovered gradually and the decision process has the following form:

$$x_0 \longrightarrow \text{observation of } d_0 \longrightarrow \text{ decision of } u_1 \longrightarrow x_1 \longrightarrow \text{observation of } d_1 \longrightarrow ...$$

$$\longrightarrow x_{T-1} \longrightarrow \text{observation of } d_{T-1} \longrightarrow \text{ decision of } u_{T-1} \longrightarrow x_T.$$

The initial state x_0 is known and the optimization variables u_t and x_t must be chosen in a non-anticipative way: u_t is seen as a stochastic process $u_t(d_{[t]})$ and x_t is seen as a stochastic process $x_t(d_{[t-1]})$. A rigourous formulation of the problem as a multistage stochastic linear program is the following:

$$\begin{array}{ll} & \min \qquad \mathbb{E}\Big(\sum_{t=0}^{T-1} c_t u_t(d_{[t]})\Big). \\ & x_{t+1}(d_{[t]}), u_t(d_{[t]}), \\ & \forall t \in \{0, ..., T-1\}, \\ \text{s.t.} & x_{t+1}(d_{[t]}) = x_t(d_{[t-1]}) + A_t u_t(d_{[t]}), \\ & B u_t(d_{[t]}) \leq b_t, \\ & E u_t(d_{[t]}) = d_t, \\ & x_{t+1}(d_{[t]}) \in X_{t+1}, \\ & \forall t \in \{0, ..., T\}, \end{array} \qquad (\mathcal{P})$$

The stochastic process d_t being discrete, problem $\mathcal{P}$ is finite-dimensional. However, since the number of variables increases exponentially with the number of stages, it cannot be solved directly.

[2]available at http://www.quantize.maths-fi.com/

Dynamic programming The SDDP algorithm uses dynamic programming in order to decompose the general problem into T sub-problems. Let us introduce two value functions $\mathcal{V}_t(x_t)$ and $V_t(x_t, d_t)$, defined recursively by, for all t in $\{0, ..., T-1\}$, for all x_t in X_t, and for all d_t in $\mathbb{R}$,

$$\mathcal{V}_T(x_T) = 0, \tag{10}$$

$$V_t(x_t, d_t) = \min_{\substack{u_t,\, x_{t+1},\, \text{s.t.} \\ x_{t+1}=x_t+A_t u_t, \\ Bu_t \leq b_t, \\ Eu_t = d_t, \\ x_{t+1} \in X_{t+1},}} c_t u_t + \mathcal{V}_{t+1}(x_{t+1}), \tag{11}$$

$$\mathcal{V}_t(x_t) = \sum_{i=1}^{K} p_t^i \, V_t(x_t, d_t^i). \tag{12}$$

In the sequel, we will also use the notation $V_t^i(x_t) = V_t(x_t, d_t^i)$, for d_t^i in D_t. By convention, we set $V_t(x_t, d_t) = \mathcal{V}_t(x_t) = +\infty$ if $x_t \notin X_t$. The value function $\mathcal{V}_t(x_t)$ stands for the expected value of the problem from t to T before the observation of d_t, given a state x_t at time t. The value function $V_t(x_t, d_t)$ is the expected value of the same problem, but knowing d_t.

Proposition 1. *The value functions have finite values for all $x_t \in X_t$ and for all $d_t \in \mathbb{R}$. Moreover, two stochastic processes $x_t(d_{[t-1]})$ and $u_t(d_{[t]})$ are optimal solutions of $\mathcal{P}$ if and only if, for all possible realizations of $d(t)$, for all t, $0 \leq t < T$, $(x_{t+1}(d_{[t]}), u_t(d_{[t]}))$ is an optimal solution to problem* (11)*.*

Proof. The dynamic programming principle is well-known, see for example [6, equations 5.2, 5.3]. The value functions $\mathcal{V}_t$ and V_t depend respectively on (x_t, d_t) and x_t only because of the stage-wise independence of the demand. Finally, the set X_t has been built in such a way that for all x_t in X_t, problem (11) is feasible, whatever the demand d_t, thanks to the slack variables. The finiteness of the value functions follows by recursion. □

A basic dynamic programming approach consists in discretizing the space state and computing successive approximations of the value functions with equations (10-12). Then, proposition 1 provides a solution for all the possible realizations of the stochastic process d_t. However, the number of discretization points increases exponentially with the dimension of the state. This fact is the well-known curse of dimensionality and prevents us from solving our problem with such an approach.

4 SDDP algorithm

General Principle The SDDP algorithm stands on Benders' decomposition, proposed initially in [2]. Van Slyke and Wets used this technique to develop the "L-Shaped" method for two-stage stochastic linear problems in [22] and Birge extended their method to multistage problems in [4]. Pereira and Pinto applied these techniques in [16] to solve a problem of optimal scheduling of a hydrothermal generating system and developed SDDP, the algorithm we used.

SDDP is an approximate dynamic programming algorithm, based on the following two key ideas.

- The value functions $\mathcal{V}_t$ being convex functions of x_t, one can compute an outer approximation of these functions by taking the supremum of a family of minoring affine functions. These affine functions are called *optimality cuts* or *Benders' cuts*.

- At each iteration, the algorithm generates a sequence of "realistic" states x_t^*. Then, cuts are computed in such a way that they provide a good approximation of the value functions in the neighborhood of x_t^*.

After k iterations, the algorithm has computed k cuts for the value function $\mathcal{V}_t$, which are denoted by $H_t^j x_t + h_t^j$, and indexed by j, $1 \leq j \leq k$. Thus, the corresponding outer approximation, denoted by $\overline{\mathcal{V}}_t$ is

$$\overline{\mathcal{V}}_t(x_t) = \max_{1 \leq j \leq k} \{ H_t^j x_t + h_t^j \} \leq \mathcal{V}_t(x_t).$$

The three steps of the k-th iteration of the algorithm are the following.

- Simulate one particular realization d_t^* of d_t.
- Forward pass: for times t from 0 to $T-1$, compute a sequence of states x_t^* which is optimal for the approximate value functions $\overline{\mathcal{V}}_t$ and for the realization d_t^*.
- Backward pass: for times t from $T-1$ to 0, compute an optimality cut $H_t^k x_t + h_t^k$, relevant in the neighborhood of x_t^*.

Let us describe precisely the forward and backward passes as well as the stopping criterion.

Forward pass Given a particular realization d_t^* of process d_t, we have to generate the corresponding solutions u_t^* and x_t^* for the approximate value functions $\overline{\mathcal{V}}_t$. Once the solutions u_0^*, x_1^*,...,u_{t-1}^*, x_t^* have been computed, u_t^* and x_{t+1}^* are solutions to the following problem:

$$\begin{array}{c} \min \quad c_t u_t + \overline{\mathcal{V}}_{t+1}(x_{t+1}), \\ u_t,\, x_{t+1},\ \text{s.t.} \\ x_{t+1} = x_t + A_t u_t, \\ B u_t \leq b_t, \\ E u_t = d_t, \\ x_{t+1} \in X_{t+1}, \end{array} \qquad (\overline{\mathcal{P}}_t(x_t, d_t))$$

for $x_t = x_t^*$ and $d_t = d_t^*$. This problem is equivalent to the following linear program:

$$\begin{array}{c} \min \quad c_t u_t + v, \\ u_t,\, x_{t+1},\, v,\ \text{s.t.} \\ x_{t+1} = x_t + A_t u_t, \\ B u_t \leq b_t, \\ E u_t = d_t, \\ x_{t+1} \in X_{t+1}, \\ \mathbf{1} v \geq H_{t+1} x_{t+1} + h_{t+1}, \end{array}$$

where v is an additional variable which stands for the expected future cost and $\mathbf{1}$ is a column vector composed of as many ones as optimality cuts. The sequences of states obtained at the different iterations can be considered as realistic in so far as the approximations are getting closer to the exact value functions $\mathcal{V}_t$. Moreover, the cuts provide a good approximation of the value function in the neighborhood of the points for which they have been computed. In a way, the algorithm concentrates on the most probable areas of the state space to improve the approximation of the value functions, where needed.

Backward pass First, let us justify the convexity of the value functions.

Lemma 2. *For all t, $0 \leq t < T$, the value functions $V_t(x_t, d_t)$ and $\mathcal{V}_t(x_t)$ are convex functions of (x_t, d_t) and x_t respectively.*

Proof. We prove this lemma by backward induction. The function $\mathcal{V}_T(x_T)$ is convex, being equal to 0. Let t, $0 \leq t \leq T$, suppose that $\mathcal{V}_t(x_t)$ is convex. Then, $V_{t-1}(x_{t-1}, d_{t-1})$ is convex, since it is the value of a convex optimization problem (11) where (x_{t-1}, d_{t-1}) appears at the right-hand-side of the constraints. Then, $\mathcal{V}_{t-1}(x_{t-1})$ is convex, as a sum of convex functions. □

Now, let us fix t, $0 \leq t < T$, and let us focus on the linear program $\overline{\mathcal{P}}_t(x_t, d_t)$. It is feasible, by construction of X_t. For convenience, the linear constraint $x_{t+1} \in X_{t+1}$ is denoted by $G_{t+1}x_{t+1} \leq g_{t+1}$. We consider the dual problem of $\overline{\mathcal{P}}_t(x_t, d_t)$, with the variables π, β, ε, γ and η.

$$\begin{array}{c} \max \\ \pi, \beta \leq 0, \varepsilon, \gamma \leq 0, \eta \leq 0, \\ \pi A_t + \beta B + \varepsilon E = c_t, \\ \eta \mathbf{1} = -1, \\ \pi + \gamma G_{t+1} + \eta H_{t+1} = 0, \end{array} \quad \pi x_t + \beta b_t + \varepsilon d_t + \gamma g_{t+1} - \eta h_{t+1}. \qquad (\mathcal{D}_t(x_t, d_t))$$

The important point in the explicit formulation of this dual problem is that the parameters x_t and d_t do not appear in the constraints. Indeed, as we already mentioned, the cost function (in the primal problem) does not depend on the state variables. We denote by $(\pi^i, \beta^i, \varepsilon^i, \gamma^i, \eta^i)$ an optimal solution to the problem for $x_t = x_t^*$ and $d_t = d_t^i$ and we denote by $W_t^i(x_t)$ the finite value of the dual problem for $d_t = d_t^i$ and for a given value of x_t.

Proposition 3. *For all x_t in X_t, the following inequalities hold:*

$$W_t^i(x_t) \geq \pi^i x_t - \pi^i x_t^* + W_t^i(x_t^*), \tag{13}$$

$$V_t^i(x_t) \geq W_t^i(x_t), \tag{14}$$

$$\mathcal{V}_t(x_t) \geq \Big(\sum_{i=1}^K p_t^i \pi^i\Big) x_t + \Big(\sum_{i=1}^K p_t^i \big(-\pi^i x_t^* + W_t^i(x_t^*)\big)\Big). \tag{15}$$

The first inequality is tight for $x_t = x_t^*$.

Proof. Let $x_t \in X_t$. As we mentioned, the constraints of the dual problem $\mathcal{D}_t(x_t, d_t)$ remain the same when the value of x_t changes. As a consequence, the point $(\pi^i, \beta^i, \varepsilon^i, \gamma^i, \eta^i)$ is a feasible point of the dual problem with $d_t = d_t^i$ whatever the value of x_t. Therefore,

$$\begin{aligned} W_t^i(x_t) &\geq \pi^i x_t + \beta^i b_t + \varepsilon^i d_t^i + \gamma^i g_{t+1} - \eta^i h_{t+1} \\ &= \pi^i x_t - \pi^i x_t^* + W_t^i(x_t^*), \end{aligned}$$

whence inequality (13). Let us prove inequality (14). $V_t^i(x_t)$ is the value of problem (11) and we know from the linear programming theory that $\overline{\mathcal{P}}_t(x_t, d_t)$ and its dual have the same value, $W_t^i(x_t)$. Problems (11) and $\overline{\mathcal{P}}_t(x_t, d_t)$ have the same constraints but the first one has a greater cost function, whence inequality (14). We finally obtain:

$$\mathcal{V}(x_t) = \sum_{i=1}^K p_t^i V_t^i(x_t) \geq \sum_{i=1}^K p_t^i W_t^i(x_t) \geq \underbrace{\Big(\sum_{i=1}^K p_t^i \pi^i\Big)}_{:=H_t^k} x_t + \underbrace{\Big(\sum_{i=1}^K p_t^i \big(-\pi^i x_t^* + W_t^i(x_t^*)\big)\Big)}_{:=h_t^k},$$

whence inequality (15). □

Proposition 3 provides an explicit method to compute a new optimality cut $H_t^k(x_t) + h_t^k$. Notice that the computation requires the resolution of the problem $(\overline{\mathcal{P}}_t(x_t, d_t))$ for the K possible values of d_t. Moreover, for the point x_t^*, the equality $H_t^k x_t^* + h_t^k = \sum_{i=1}^K p_t^i W^i(x_t^*)$ holds, which means that the new cut has the highest possible value in x_t^*, given the current approximation of $\mathcal{V}_{t+1}$.

Figure 2 illustrates how the T sub-problems exchange information during the forward and backward passes. The primal problem associated with stage t passes on the "next state" x_{t+1}^* to stage $t+1$ whereas the dual problem passes on an optimality cut to stage $t-1$.

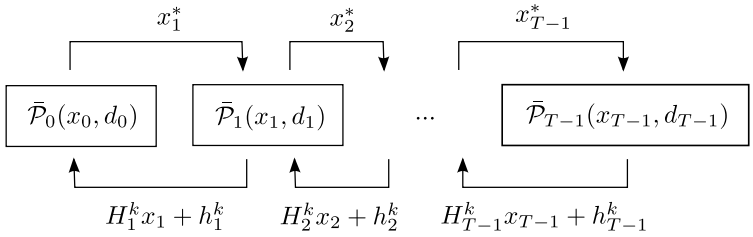


Figure 2: A forward pass and a backward pass

Stopping criterion We use the criterion described in [16]. It consists in computing a lower bound and an upper one of the value $\mathcal{V}_0(x_0)$ of problem $\mathcal{P}$. A lower bound $\underline{v}$ is directly obtained with $\overline{\mathcal{V}}_0(x_0)$. A relevant upper bound of $\mathcal{V}_0(x_0)$ could be the mean cost of the strategy defined by the approximate value functions $\overline{\mathcal{V}}_t$. To compute this bound, we should solve the problems $\overline{\mathcal{P}}_t(x_t, d_t)$ for all the possible scenarios of demand d_t, which is impossible. Instead, we can compute a confidence interval of this upper bound with a Monte-Carlo method. We fix a number N sufficiently large, we simulate N realizations of d_t. We solve the associated approximate solutions and costs with problems $\overline{\mathcal{P}}_t(x_t, d_t)$. We denote respectively by $\overline{v}$ and σ the mean value and the standard deviation of the N obtained costs. The value $\overline{v}$ is called forward cost. We compute the 95% confidence interval given by

$$\left[\overline{v} - \frac{1,96\sigma}{\sqrt{N}}, \overline{v} + \frac{1,96\sigma}{\sqrt{N}}\right].$$

Then, the criterion is the following: stop the algorithm when $\underline{v}$ belongs the confidence algorithm. This test must be regularly performed during the algorithm.

Convergence It is proved in [6, chapter 7, theorem 1] and [17] that the SDDP algorithm converges in a finite time to an optimal solution. The proof relies on the fact that there is only a finite number of optimality cuts. See also [1] for a proof of convergence in a more general case and [20] for a statistical analysis of the convergence of the algorithm and a criticism of the above stopping criterion (remark 1).

5 Review of computational improvements

Stochastic programming techniques have been widely studied in literature and numerous enhancements have been designed for various stochastic problems. In this section, we give a short review of some of the enhancements that were proposed. We do not aim at being exhaustive but we try to confront them to our particular problem. We refer to [6] for a survey on these techniques.

Similarities of sub-problems During the backward pass, at stage t, one has to solve K very similar problems, since the constraint of equilibrium between supply and demand, $Eu_t = d_t$ is the only one to change. Techniques have been proposed to reduce the number of simplex pivots required to solve linear programs differing only on the right-hand-side of the constraints . We refer again to [6, section 5.6] for more details.

For our application, we used a commercial solver, CPLEX. The software includes efficient techniques of "hot starts", reducing considerably the number of simplex pivots. Roughly speaking, in the backward pass, when the value of d_t changes, the solver initializes the simplex algorithm with the previous optimal simplex basis.

Initialization of the algorithm For our problem, the stochastic data appear linearly at the right-hand side of the constraints only. As we mentioned in lemma 2, the intermediate value functions $V_t(x_t, d_t)$ are convex with respect to x_t and d_t simultaneaously. In this case, one can obtain a lower bound of the value functions $\mathcal{V}_t(x_t)$ by solving first the problem where the demand is taken equal to its mean value $\sum_{i=1}^{K} p_t^i d_t^i$. The problem obtained is deterministic and can be accurately solved with forward and backward passes. Denoting by $\widetilde{\mathcal{V}}_t(x_t)$ the associated value functions, the following inequality:

$$\widetilde{\mathcal{V}}_t(x_t) \leq \mathcal{V}_t(x_t), \quad \forall t,\ 0 \leq t \leq T,\ \forall x_t,$$

holds and can be shown by backward recursion, by using Jensen's inequality. Therefore, the optimality cuts computed for the deterministic problem remain valid for the stochastic problem. These cuts enable us to reduce the number of iterations needed to reach convergence. A complete proof of this property can be found in [3]. The technique was also used successfully in [11].

For our application, this initialization proved efficient and allows an important decreasing of the forward cost.

Sequencing protocols The algorithm we described is known as the "fast-forward, fast-back" procedure. Roughly speaking, one iteration of our algorithm goes from stage 0 to stage $T-1$ in the forward pass, then it goes from stage $T-1$ to stage 0 in the backward pass. The "forward first" procedure goes from stage t to stage $t-1$ only when the solutions of the problem for stages t,..., $T-1$ are optimal. The "backward-first" procedure goes from stage t to stage $t+1$ only when the solutions of the problem for stages 0,..., t are optimal (for the current approximation of the value function at time t).

We followed the conclusions of the numerical results of Gassmann [13], showing that the fast-forward fast-back procedure was more efficient.

Sampling The SDDP algorithm belongs to the broad class of sampling algorithms in so far as it works with only a small part of the set of all the possible realizations of the stochastic process d_t. The scenarios involved are those obtained after the forward pass. In [11], Donohue and Birge developed the Abridged Nested Decomposition algorithm, which uses more representative realizations of the stochastic process in the forward pass. Let us also mention the Cutting-Plane and Partial-Sampling algorithm of Chen and Powell [9] and the Dynamic Outer Approximation Sampling Algorithms of Philpott and Guan [17], which use sampling to avoid the resolution of all the K subproblems at each stage in the backward pass.

We have not tested these techniques.

Multicuts This technique was proposed by Birge and Louveaux in [7]. In our description of the algorithm, we solve in the backward pass all the problems associated with the different values

of d_t, to obtain an optimality cut of the value function $V_t^i(x_t)$ for all i, following proposition 3. Then, we only keep the mean value of these cuts, which is a cut for the value function $\mathcal{V}_t(x_t)$. On the contrary, the idea of the multicut algorithm is to keep in memory all the intermediate cuts. We denote them by $H_t^{j,i}x_t + h_t^{j,i}$, the new problems to solve are the following:

$$\begin{array}{c} \min \\ \scriptstyle u_t, x_{t+1}, v \\ \scriptstyle x_{t+1}=x_t+A_tu_t, \\ \scriptstyle Bu_t\leq b_t, \\ \scriptstyle Eu_t=d_t, \\ \scriptstyle x_{t+1}\in X_{t+1} \\ v^i \geq H_{t+1}^i x_{t+1} + h_{t+1}^i, \ \forall i \end{array} \quad c_t u_t + \sum_{i=1}^{K} p_t^i v^i. \tag{16}$$

With this method, we obtain a better approximation of the value functions. On the other hand, the number of optimality cuts is multiplied by K.

This technique could not be tested for our application problem, given that it requires a too wide space memory. On the contrary, we tried to limit the number of optimality cuts.

Parallelization SDDP can be efficiently parallelized. A first approach consists in realizing several forward and backward passes on different processors. Another approach consists in partitioning the time period. Each processor solves the problems associated to the stages of one subset of the partition. It is then possible to launch a new iteration of the algorithm even if the previous one is not finished yet. We refer to [5, 19] for details.

We have not tested such techniques.

6 Pruning optimality cuts

While the algorithm is running, the number of optimality cuts increases. Optimality cuts computed during the first iterations of the algorithm can quickly become obsolete, especially the ones associated with the first stages, because they are obtained with a very bad approximation of the value functions. The accumulation of cuts can be penalizing for two reasons. First, if the number of iterations to reach the convergence is high, one risks exceeding the capacity of the storage space. Moreover, it slows down the solving of linear programs. In this section, we deal with the question of picking out the optimality cuts which can be considered as relevant.

Given an approximation $\overline{\mathcal{V}}_t(x) = \max_k H_t^k x + h_t^k$, we say that one cut j is *useless* if for all x in X_t,

$$H_t^j x + h_t^j \ \leq \ \max_{k,k\neq j} [H_t^k x + h_t^k], \tag{17}$$

or, equivalently, if

$$\overline{\mathcal{V}}_t(x) = \max_{k,k\neq j} [H_t^k x + h_t^k]. \tag{18}$$

If a given cut j is useless, we can remove it from the set optimality cuts without without damaging the approximation of the value function $\overline{\mathcal{V}}_t$. On figure 3, cut 1 is useless, while the others are useful.

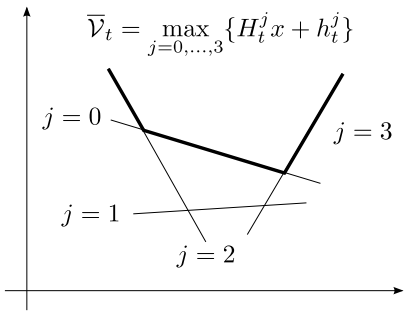


Figure 3: Cut 1 is useless.

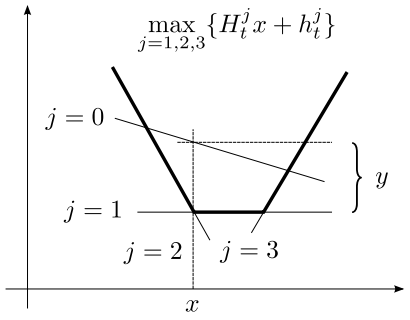


Figure 4: Test of usefulness for cut 0.

Test of usefulness A first proposition to determine if a cut j is useless consists in solving the following linear program, that we call *test of usefulness*:

$$\max_{x\in X_t} \left[H_t^j x + h_t^j - \max_{k,k\neq j} [H_t^k x + h_t^k]\right] \tag{19}$$

$$= \max_{x\in X_t} \min_{k,k\neq j} [H_t^j x + h_t^j - H_t^k x - h_t^k] \tag{20}$$

$$= \max_{\substack{x\in X_t, y\in\mathbb{R} \\ y \leq (H_t^j - H_t^k)x + (h_t^j - h_t^k), \\ \forall k, k\neq j}} y. \tag{21}$$

The cut is useless if and only if the solution y is non-positive. It should then be deleted. On the contrary, if the value of y is high, it means that the examined cut widely contributes to the approximation of the value function. Notice that in our case, X_t is bounded and thus, the problem has a finite value. Figure 4 shows the solution (x, y) to the linear program for cut 0.

The test of usefulness provides us with an exact method to determine if a cut is useless, with one linear program. However, it is impossible to perform this test for all the optimality cuts at each iteration of the algorithm, this would take too much time.

Territory algorithm In [15], the value of the test of usefulness is described as the "importance metric" (see their problem (10), which is equivalent to our problem (21)). Let us mention that in [14, section 6], the number of cuts to select is fixed and the problem of pruning is formulated as a combinatorial optimization problem. Different methods are tested.

Our approach here is different, in so far as we do not fix the number of selected cuts. Our *territory algorithm* enables us to select the cuts which are potentially useless. In the test of usefulness, we test for all the points of X_t if a given cut is above or under the other ones. The idea here is to perform the same kind of test for a discrete set of points of X_t. The method consists in associating with each cut j a set of points of X_t, denoted by P_t^j such as for all x in P_t^j, for all cut k, $H_t^j x + h_t^j \geq H_t^k x + h_t^k$. This set of points is what we call a *territory*: an area of X_t where cut j is higher than the other ones. When a cut j is computed for a point $\overline{x}_t$, the algorithm is the following:

- Set $P_t^j = \{\overline{x}_t\}$.
- Search for the cut k which maximizes $H_t^k \overline{x}_t + h_t^k$. If $H_t^k \overline{x}_t + h_t^k > H_t^j \overline{x}_t + h_t^j$, then do:
  - $P_t^j \leftarrow \emptyset$

  - $P_t^k \leftarrow P_t^k \cup \{\overline{x}_t\}$

- For all cut k, for all x in P_t^k, if $H_t^j x + h_t^j > H_t^k x + h_t^k$, then do:

  - $P_t^j \leftarrow P_t^j \cup \{x\}$
  - $P_t^k \leftarrow P_t^k \backslash \{x\}$

- If for some cut k, $P_t^k = \emptyset$, then remove cut k from the set of optimality cuts.

This heuristic selects the same cuts as the *Level 1 strategy* of de Matos *et al.* [10, section 4.2].

It is clear from the definition of a useless cut that if the territory algorithm is performed, a useless cut j has an empty set of points P_t^j and is directly deleted. However, some useful cuts can also be suppressed by the algorithm. This happens if for a given cut j,

$$\begin{aligned} &\text{for all } k \neq j, \text{for all } x \text{ in } P_t^k,\ H_t^j x + h_t^j < H_t^k x + h_t^k, \\ &\text{there exists } x \text{ in } X_t \text{ such as } H_t^j x + h_t^j \geq \max_{k,k\neq j} [H_t^k x + h_t^k]. \end{aligned}$$

However, in such a situation, it is possible that the algorithm computes again a cut which has been eliminated previously, with another point associated. As a consequence, the territory algorithm does not prevent SDDP from converging. Notice that the total number of points in the territories P_t^j increases throughout the algorithm, therefore, the test becomes more and more precise.

The advantages of this method are the following:

- the selection of the cuts to be removed is very simple and does not take much time
- the method focuses on cuts which are "useful" in a realistic area of X_t, since the points of the territories are obtained with forward passes.

Figure 5 shows optimality cuts with their territories. In figures 6 and 7, a point x of the territory of cut 1 is transfered to another cut, cut 3. In the case illustrated by figure 7, if x was the only point of the territory of cut 1, the territory algorithm would eliminate this cut, whereas it contributed to the approximation of the value function, represented in bold on the figure.

A combination of the test of usefulness and the territory algorithm A variant of the territory algorithm has been implemented, combining the territory algorithm and the test of usefulness. The idea is to refine the territory algorithm by saving from elimination some potential non-useless cuts with the test of usefulness. When the set of points P_t^j of a cut j is empty, the suggested method is the following:

- Apply the test of usefulness to cut j and denote by (x, y) its solution.

  - If $y \leq 0$, delete the cut, it was actually useless.
  - If $y > 0$, keep the cut and set: $P_t^j = \{x\}$, so that it has a new territory.

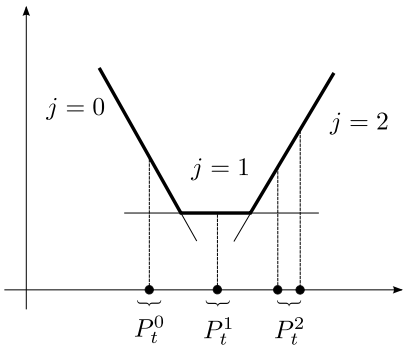


Figure 5: Cuts and territories

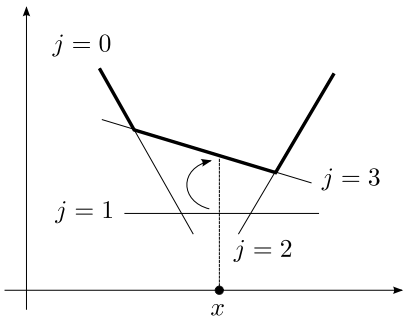


Figure 6: Transfer of a point x from P_t^1 to P_t^3

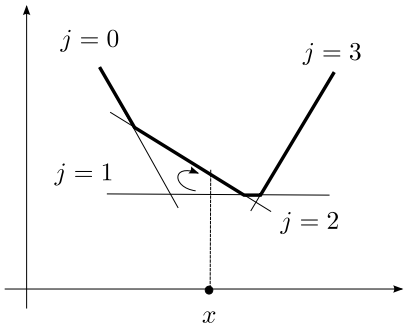


Figure 7: Elimination of a non-useless cut, cut 1

7 Numerical results

7.1 Case study

The main features of our case study are presented in table 1. The program used is $C\sharp$, the solver

Dimension of the state x_t	19
Number of zones	1
Number of storages	1
Number of contracts	18
Number of time steps	365
Number of realizations of d at time t	10

Table 1: Case study

used is Cplex v12.2 and the tests have been performed on a pi-processor computer with a 2.13 GHz quad-core CPU and 16 GByte main memory. In the numerical tests that we have realized, we compare three methods. In the first, there is no selection of the cuts. In the second, the territory algorithm is performed and in the third, the combination of the territory algorithm and the test of usefulness is performed. See pages 20 and 21 the different graphs and tables for the numerical results.

Note that we distinguish an optimization phase and a simulation phase. The algorithm is run during the optimization phase. No convergence test is realized during this phase. The optimization phase takes into account the time needed to select the cuts. During the simulation phase, we compute the solutions associated with 1000 different scenarios of demand. Like convergence tests, this phase is only composed of forward passes.

7.2 Comparison of methods of selecting cuts, without initialization of the algorithm

Optimality cuts Figure 10 shows the evolution of the number of affine minorants. As was expected, the number of cuts is the smallest with the territory algorithm and the largest when there is no selection of the cuts. Figure 12 shows the mean number of cuts per time step after the 500 iterations of the algorithm. More precisely, figure 11 represents the number of cuts for each time step of the problem. We observe that when there is a selection, the number of cuts is directly linked to the size of the area of the state space which has been explored. The territory algorithm enables to divide by 10 the number of cuts, yet a small loss of information has to be observed since the forward cost is slightly higher than the forward cost when no selection is realized. Figure 13 shows the forward cost with respect to the number of hyperplanes for the three methods. *It appears that the territory algorithm provides the most efficient representation of the value function, with the smallest number of cuts.*

Forward cost Figure 14 shows the evolution of the forward cost when no initialization is performed. The number of iterations of the algorithm is fixed to 500. *The forward cost is decreasing at similar rates for the three methods.*

Computation time Table 15 shows the time needed to run each of the three methods. The length of the optimization phase is of the same order for the territory algorithm and when there is no selection. On the contrary, the combination of the territory algorithm and the test of usefulness takes much time, since the selection of cuts requires the solving of many linear problems. *The combination of the territory algorithm and the test of usefulness is inefficient during the optimization phase.*

The length of the simulation phase is directly linked to the number of hyperplanes, since there are only forward passes. *Consequently, the simulation phase is much shorter with the territory algorithm than with the other methods.*

7.3 Results with an initialization of the algorithm

We have realized the same tests with an initialization of the algorithm. The initialization phase includes 100 iterations with a demand equal to its mean value. After this phase, we realize 400 iterations of the algorithm with the classical method.

Optimality cuts Figure 16 shows the evolution of the number of optimality cuts, which is close to the number of optimality cuts when there is no initialization. Figure 17 shows the number of cuts at each time step. Figure 18 shows the mean number of cuts per time step. As shown by figure 19, *the territory algorithm, used with an initialization still provides the most efficient representation of value function.*

Forward cost Figure 20 shows the evolution of the forward cost. After 500 iterations, the difference between the forward and the backward cost is about five times smaller with an initialization as without, as shown by figure 9 (the backward cost being equal to approximately 12 350). Note that it is normal that the cost does not decrease after each iteration, since the forward cost is only an approximation of the real cost of the strategy associated with the computed value functions. *The initialization is very efficient and permits to start with smaller values, for the three methods.*

Computation time The times of computation associated with the optimization and the simulation phases are shown in table 21. *The computation times are similar to the case without initialization, whereas much more accurate solutions have been obtained.*

Note that the initialization phase is not sufficient to solve the stochastic problem, since a large area of the space state is covered by the different possible trajectories. See figure 8, which shows the trajectories followed by the stock of gas for different scenarios, after the solving of the problem.

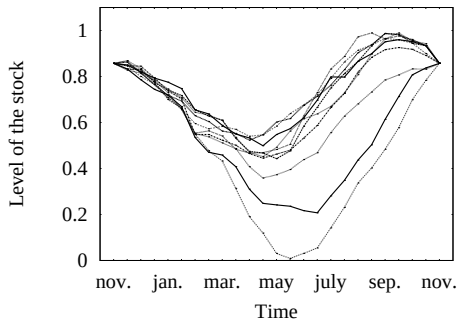


Figure 8: Level of stock for different scenarios

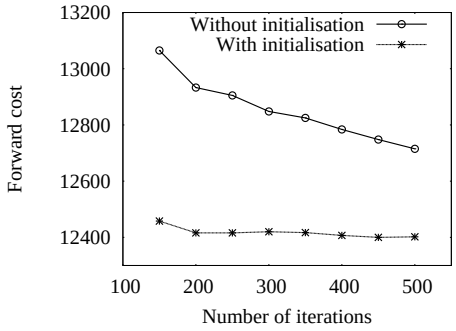


Figure 9: Forward cost with and without initialization

8 Conclusion

We have presented a gas portfolio management problem, solved with the SDDP algorithm. We have focused on methods of selecting the most relevant cuts. We have proposed a territory algorithm which eliminates useless cuts and an exact test of usefulness, which is tractable (but inefficient) when associated with the territory algorithm.

The numerical results show first the great advantage of using the initialization of the algorithm described section 5. With this technique, the forward cost decreases very quickly at the first iterations, and we can obtain more accurate solutions to the problem.

Secondly, the convergence of the forward cost is similar for three methods (without selection, the territory algorithm and the combination with the test of usefulness), whether or not there is an initialization. It appears that the selection of cuts has only a small impact on the value of forward costs and on the time of the optimization phase. On the contrary, the number of hyperplanes can be divided by 10 with an efficient selection, like the territory algorithm, which is of great interest if a long simulation phase is performed or if many forward passed are performed during convergence tests. Let us recall also that selecting cuts allows to avoid memory problems when computing accurate solutions.

Results without initialization of the algorithm

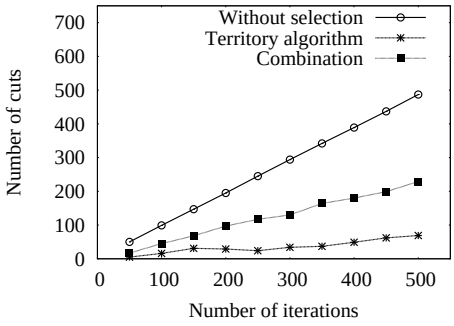


Figure 10: Evolution of the mean number of optimality cuts

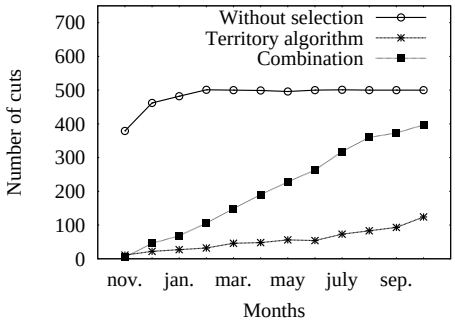


Figure 11: Number of optimality cuts at each time step

Without selection	490
Territory algorithm	220
Combination of territory algorithm and test of usefulness	55

Figure 12: Mean number of cuts per time step after 500 iterations

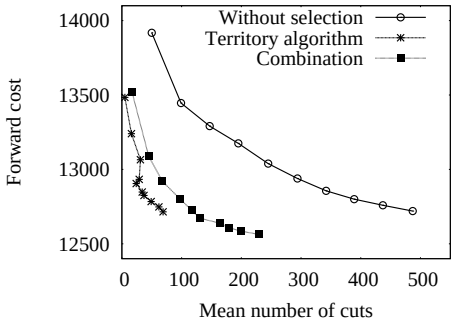


Figure 13: Forward cost with respect to the mean number of cuts

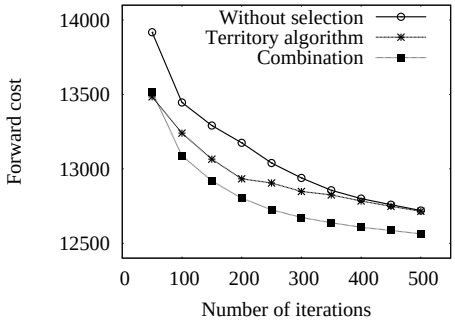


Figure 14: Evolution of the forward cost with respect to the number of iterations

	Optimization phase	Simulation phase
Without selection	543 s	798 s
Territory algorithm	690 s	168 s
Combination of territory algorithm and test of usefulness	2 181 s	382 s

Figure 15: Computation time during optimization and simulation phases

Results with an initialization of the algorithm

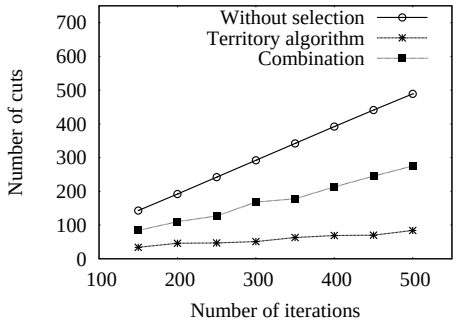


Figure 16: Evolution of the mean number of optimality cuts

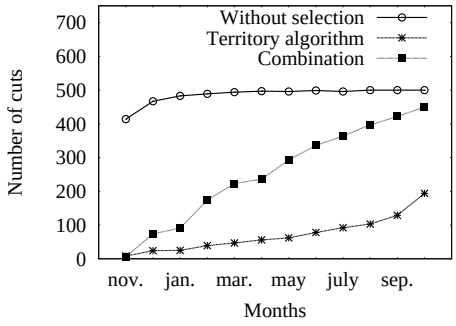


Figure 17: Number of optimality cuts at each time step

Without selection	490
Territory algorithm	275
Combination of territory algorithm and test of usefulness	84

Figure 18: Mean number of cuts per time step after 500 iterations

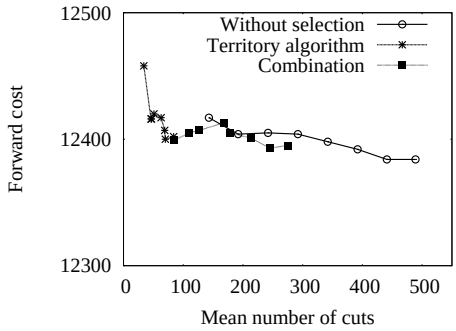


Figure 19: Forward cost with respect to the mean number of cuts

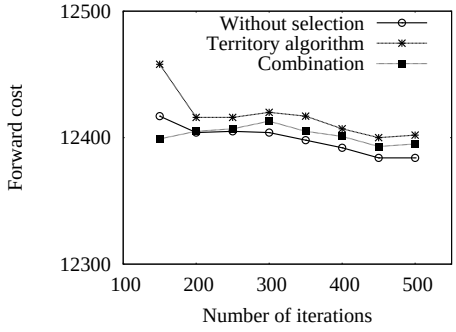


Figure 20: Evolution of the forward cost with respect to the number of iterations

	Optimization phase	Simulation phase
Without selection	772 s	649 s
Territory algorithm	792 s	205 s
Combination of territory algorithm and test of usefulness	2 090 s	408 s

Figure 21: Computation time during optimization and simulation phases

References

[1] Kengy Barty. A note on the convergence of the SDDP algorithm. Preprint, July 2012.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Comput. Manag. Sci.*, 2(1):3–19, 2005. Reprinted from Numer. Math. **4** (1962), 238–252 [MR0147303].

[3] John R. Birge. The value of the stochastic solution in stochastic linear programs with fixed recourse. *Math. Programming*, 24(3):314–325, 1982.

[4] John R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Oper. Res.*, 33(5):989–1007, 1985.

[5] John R. Birge, Christopher J. Donohue, Derek F. Holmes, and Oleg G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Math. Programming*, 75(2, Ser. B):327–352, 1996. Approximation and computation in stochastic programming.

[6] John R. Birge and François Louveaux. *Introduction to stochastic programming*. Springer Series in Operations Research. Springer-Verlag, New York, 1997.

[7] John R. Birge and François V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European J. Oper. Res.*, 34(3):384–392, 1988.

[8] J. Frédéric Bonnans, Zhihao Cen, and Thibault Christel. Energy contracts management by stochastic programming techniques. *Annals of Operations Research*, 200:199–222, 2012.

[9] Z. L. Chen and W. B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *J. Optim. Theory Appl.*, 102(3):497–524, 1999.

[10] Vitor L. de Matos, Andy B. Philpott, and Erlon C. Finardi. Improving the performance of stochastic dual dynamic programming. Preprint, July 2012.

[11] Christopher J. Donohue and John R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Oper. Res.*, 1(1):20–30, 2006.

[12] Wendell H. Fleming and William M. McEneaney. A max-plus-based algorithm for a Hamilton-Jacobi-Bellman equation of nonlinear filtering. *SIAM J. Control Optim.*, 38(3):683–710 (electronic), 2000.

[13] Horand I. Gassmann. MSLiP: a computer code for the multistage stochastic linear programming problem. *Math. Programming*, 47(3, (Ser. A)):407–423, 1990.

[14] S. Gaubert, W. McEneaney, and Zheng Qu. Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 1054 –1061, dec. 2011.

[15] W.M. McEneaney, A. Deshpande, and S. Gaubert. Curse-of-complexity attenuation in the curse-of-dimensionality-free method for HJB PDEs. In *American Control Conference, 2008*, pages 4684 –4690, june 2008.

[16] M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Math. Programming*, 52(2, Ser. B):359–375, 1991.

[17] A. B. Philpott and Z. Guan. On the convergence of stochastic dual dynamic programming and related methods. *Oper. Res. Lett.*, 36(4):450–455, 2008.

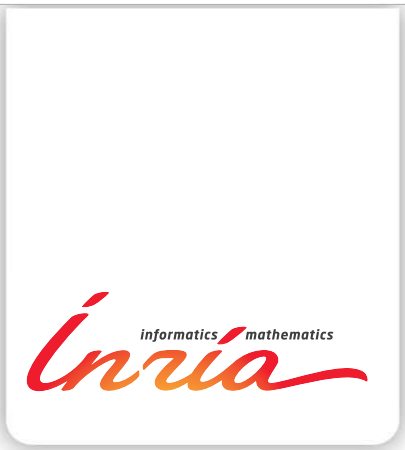
[18] Andrzej Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Math. Programming*, 35(3):309–333, 1986.

[19] Andrzej Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Math. Programming*, 58(2, Ser. A):201–228, 1993.

[20] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63 – 72, 2011.

[21] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming*, volume 9 of *MPS/SIAM Series on Optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009. Modeling and theory.

[22] R. M. Van Slyke and Roger Wets. *L*-Shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math.*, 17:638–663, 1969.



RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399